CSCI 3353 Object Oriented Design

Homework Assignment 1
Due Monday, January 29

Download the file *DiningHallSimulation.java*. This program performs a simulation of a dining hall. It makes the following assumptions:

- The dining hall has four cash registers.
- Each second, there is a 15% chance that a customer will be ready to check out.
- A customer will always enter the shortest checkout line.
- Each customer has between 1 and 10 items. Each item takes two seconds to scan, and payment takes another 10 seconds. Thus if a customer has n items, it will take 2n+10 seconds to check out.

The simulation runs for 100,000 seconds. Afterwards, it prints the number of customers serviced and the average wait time per customer for each register.

This code does its job reasonably elegantly. However, it has a fundamental design flaw: It is organized in a non-object-oriented way. There is only one class. The purpose of this assignment is to refactor it using the best object-oriented style you know. In particular, I want you to focus on creating modular, well-designed classes. In particular, your design should satisfy the *Most Qualified Class* rule, the *Encapsulation* rule, and the *Low Coupling* rule to the extent possible.

To simplify the grading, I would like everyone's solution to consist of the following four classes: *Simulation*, *DiningHall*, *Customer*, and *CashRegister*. The class *Simulation* controls the simulation loop and contains the main method. The scope of the other classes should be apparent.

Your task is to address the following questions: What is the responsibility of each class? What methods should each class support? What work should each method do? How do the classes interact?

I do not want you to come up with a new way to do the simulation. Your revised code should generate the same output as before. Don't try to make the code more efficient. All I want you to do is refactor the existing code – that is, to find a way to do essentially the same thing, but in a more object-oriented way. To do this, you will need to **thoroughly** understand what my code does; otherwise, you will not be free enough to make the necessary changes. If you have any questions, please ask me.

WARNING: This assignment is not as simple as it might seem. The hard part is to figure out what responsibilities to assign to each class. This is the sort of thing that you need to mull over, which means that it is unlikely you will come up with a really good solution in a single sitting. Think about what makes my code so poorly designed, and then figure out how to do it better. You might be surprised at the dramatic change that a good refactoring can produce.

HINT: The *arrivalTimes* and *serviceTimes* lists hold a key to the solution. What information do they contain? How can you represent the same information in a more object-oriented way?

WHAT TO SUBMIT:

You should submit five files: four java files and a pdf file. Submit these individually. Do NOT zip them into a single file.

The java files should be named *Simulation.java*, *DiningHall.java, Customer.java*, and *CashRegister.java*. They should all be in the package *hw1*.

The pdf file should be named *SimulationClassDiagram.pdf*. It should contain the class diagram for your solution: that is, it should have a box for each class, with the name of the class at the top and the declarations of its public methods below, and arrows denoting the client-service relationships among the classes.

I don't care how you create the diagram. One possibility is to draw the diagram (neatly!) on paper. There are a couple of ways to convert the diagram to pdf. If you have access to a scanner, you can scan the paper to a pdf file. Or you can use your phone to take a photo of the diagram, and then download the photo to your computer, insert it into a Word document, and save the document as pdf.

If you could not get something to work correctly, explain the issue in a text document and submit that also.